LEVEL ~~II~~

ESD-TR-79-251

(10) B.S.

AD A077874

CONDUCTING AN INDEPENDENT REVIEW GROUP (IRG) EVALUATION OF C^3I SOFTWARE

Daniel R. Baker, Captain, USAF
David A. Herrelko, Captain, USAF
Charles J. Grewe, Jr., Lt. Colonel, USAF
Directorate of Computer Systems Engineering
Electronic Systems Division
Hanscom AFB, MA 01731

August 1979

Prepared for

DEPUTY FOR TECHNICAL OPERATIONS
ELECTRONIC SYSTEMS DIVISION (AFSC)
HANSCOM AIR FORCE BASE, MA 01731

79 12 10 008

[illegible]
JAMES W. NEELY, JR., LT COL, USAF
Acting Chief, Technology Applications Division
Deputy for Technical Operations

[illegible]
WILLIAM J. LETENDRE, GS-14
Deputy Director, Computer Systems Engineering

[illegible]
NORMAND MICHAUD, Colonel, USAF
Director, Computer Systems Engineering

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. ~~REPORT NUMBER~~ (14) ESD-TR-79-251	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. ~~TITLE (and Subtitle)~~ (6) Conducting an Independent Review Group (IRG) Evaluation of C^3I Software		5. TYPE OF ~~REPORT & PERIOD~~ COVERED (9) Final [redacted] Rept.,
		6. ~~PERFORMING ORG. REPORT NUMBER~~
7. ~~AUTHOR(s)~~ (10) Daniel R. Baker, Captain, USAF David A. Herrelko, Captain, USAF Charles J. Grewe, Jr., Lt Colonel, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. ~~PERFORMING ORGANIZATION NAME~~ AND ADDRESS Directorate of Computer Systems Engineering (TOI) Electronic Systems Division (AFSC) Hanscom AFB, Bedford, MA 01730		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE28010F Project 642260
11. CONTROLLING OFFICE NAME AND ADDRESS See Item 9		(11) 12. REPORT DATE Aug[redacted]079
		13. ~~NUMBER OF PAGES~~ 44
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* (12) 47		15. SECURITY CLASS. *(of this report)* Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Independent Review Group
Evaluation Criteria
Software Development

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report provides guidelines and procedures for conducting Independent Reviews of Air Force software systems acquisition programs. Experiences of one recent Independent Review are used as the basis for many of the examples, procedures, and recommendations of this report. The Independent Review process is discussed in detail from the formation of the Independent Review Group (IRG) to preparing the final report. Examples of IRG data collection forms are provided.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

127 100

CONTENTS

SECTION 1

1.0 INTRODUCTION

In January 1979, an Independent Review Group (IRG) from the Electronic Systems Division (ESD) was successful in analyzing and evaluating the applications software development for one of the ongoing system acquisition programs in the Air Force. The success of that IRG was due to many factors: the way the IRG was managed, the composition and selection of team members, the technical approach taken to review the software, the manner in which the findings were presented, and the form of the documentation. Prior to this review, no handbook existed to assist the IRG. Instead, the IRG developed tools and techniques as needed from day to day. Based upon this experience, this paper describes the independent review process and offers guidance and tools to simplify the work of analyzing the status future software development activity.

SECTION 2

2.0 PREPARING FOR THE REVIEW

The independent review process usually begins after problems of cost, schedule, or performance have attracted senior level management attention to a troubled program. The nature of IRG investigations is that the program is seldom in a healthy condition. In the case of the referenced IRG, the program manager identified the software as a high risk element, because it could not pass Preliminary Qualification Tests (PQT). As a result, the program manager requested an IRG investigation of the software.

2.1 DEFINING THE PROBLEM

Figure 1 depicts the independent review process from initial client (program office) contact to publication of the final report. The IRG may be tasked at any phase of program development and to review any portion of software in the system. The IRG may also be tasked to evaluate software management areas, such as configuration control, test discipline, or manpower. In any case, an IRG must always be created to answer specific questions and limited in scope to the immediate problem. The types of questions that could be asked about the software development are:

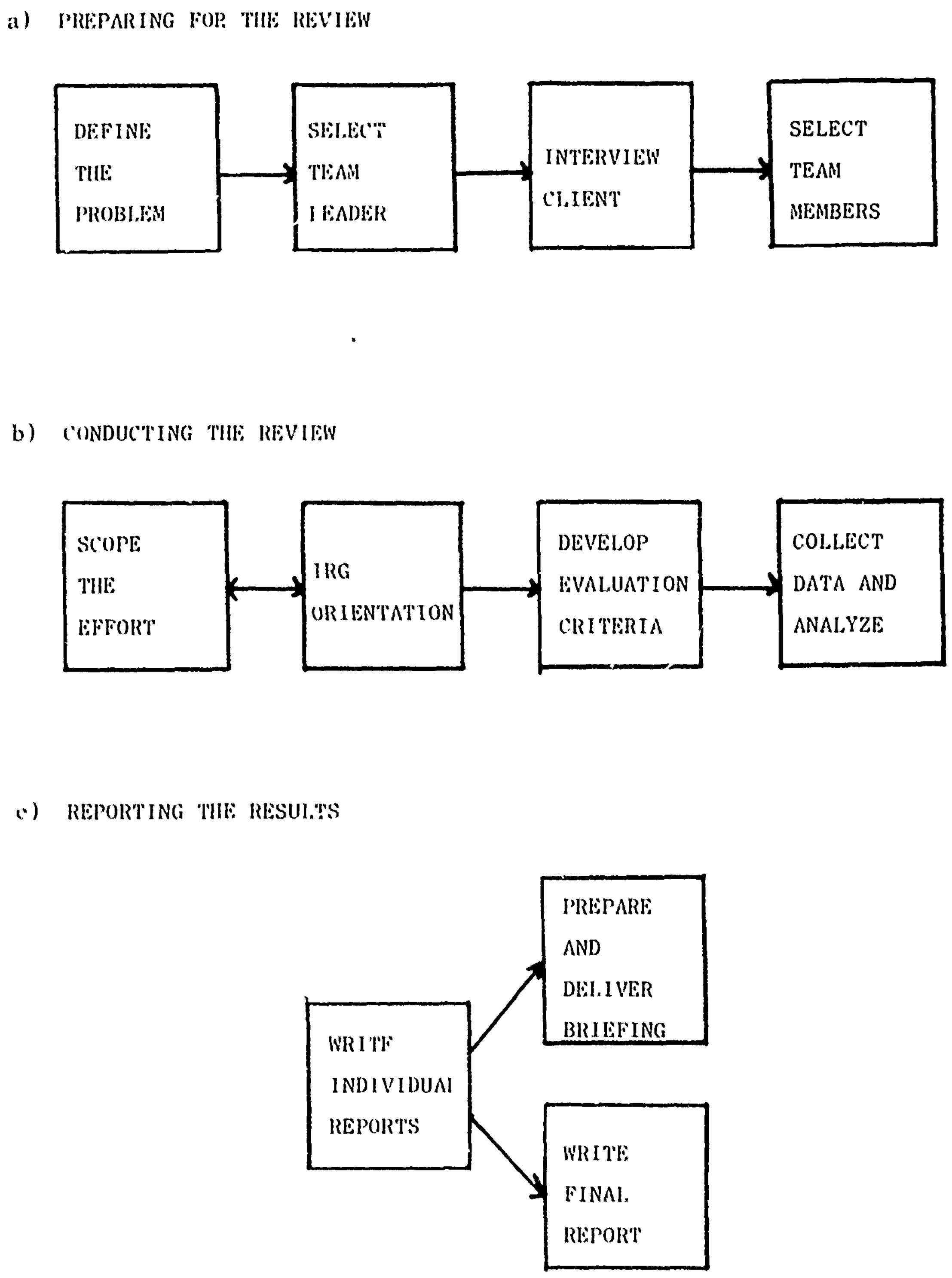

FIGURE 1. THE INDEPENDENT REVIEW PROCESS

a) Given the current software status, can the software perform the required tasks?

b) What needs to be accomplished right now before the software can proceed in its development?

c) What change in direction must the software development take before it can be delivered to the field as operable, reliable, maintainable code?

These questions are typically asked about software which is not working. They could apply to software in any phase of development. The technical skills and information needed to answer these questions will differ for each phase of development, and for each program.

2.2 TEAM COMPOSITION

The team composition is by far the most critical element of any IRG. The wheels of system acquisition grind in real time, and IRGs do not have the luxury of months to reflect on all the fine points they might wish to consider. Therefore,

a) the team must be competent

b) the team must have a leader

c) the team must understand its mission

d) the team must move fast, and work full-time on the IRG

e) the team must not waste time re-inventing administrative and technical review procedures that have already been worked out by other IRG teams.

2.2.1 THE TEAM LEADER

The team leader should have the authority to select the team, contract the exact type of investigation, clarify any constraints, establish a schedule, assign responsibilities, make progress reports, direct the technical secretary, obtain clerical support, edit and sign thank-you letters, and approve or make all formal presentations required. Since an IRG effort is short and intense, the leader will usually work long hours recording and planning each day. He must be evaluating the team's results, as they occur, so that an IRG recommendation can be made as rapidly as possible. He should have the final say on the contents of the executive and technical summaries, and present briefings to the program office, contractors, and others as required.

The following team leader qualifications are required in order to respond to the above responsibilities. He must have several years of experience as a software manager. He should be able to communicate the political and technical issues confronting the team members. He must deal with authority with the program office (PO) and contractor management. He must be current in the procurement process and have working level experience with the particular stage of software development under study. Since the leader will be the only IRG member to witness and review the entire IRG effort (the other members will be too busy with specific, specialized tasks), he must have a corresponding system level view of the IRG activities to scope the IRG results to the overall PO software development effort.

Once selected, the leader should interview his client, to formally determine the purpose of the review and any constraints on his work. Following the meeting, the leader should draft a memorandum for record documenting the understanding. There must be a contract between client and team leader, to avoid misdirections and false starts, and to insure that the purpose of the review is understood and the product to be delivered is the right one.

2.2.2 THE TECHNICAL STAFF

The rest of the team must be comprised of technical experts, who have had considerable and direct experience with software development. The IRG team should be tailored to the job. For example, the referenced IRG was required to evaluate software program listings in assembly and higher order language (HOL). The individual team members were required to learn the language, evaluate a great quantity of code, and come to a conclusion in less than four days. This type of effort requires a very experienced programming group. However, just as much experience (of perhaps a different type) is necessary with any phase of software development. For instance, during a preliminary design, the team would have to be capable of evaluating designs, tracing requirements, understanding specifications, and evaluating hardware/software/firmware tradeoffs. This may require system analysis, rather than programming, experience.

In addition to competence, the team must be mature. The PO and the contractor under scrutiny will all be ill-at-ease about the IRG process, and possibly resentful of the extra attention, interference, and potential work stoppage. The presence of a technically sound, but unseasoned team member may prejudice an otherwise successful investigation. One thoughtless comment may cause a contractor claim against the government, or may reveal an 'Air Force Only' piece of information that will affect a source selection or other procurement activity.

By definition, the IRG should consist of people who are independent of the activities under review. These individuals will not be familiar with the software or the system, and will require orientation and technical assistance by the PO as the review progresses. A few auxiliary team members who have knowledge of the system can be helpful for answering questions. However, caution must be taken here. If these individuals have a vested interest in the IRG outcome, they will tend to 'explain away' the bad software, and create a continual disruption as the team works. Whether these auxiliary members are from the PO or the contractor, their limited role should be spelled out before the review begins.

The team must function smoothly, with a minimum of time lost in disagreements among members. The team leader has the final say on all matters, including job assignment, and the individuals must be willing to accept any job assigned. The IRG requires a non-hierarchical structure that focuses on skills rather than rank. This structure must be formed out of the system requirements, system design particulars, and problem areas suspected, rather than by the dictates of an organizational chart or personal ambition.

The team must be of a manageable size. The team referenced had eight technical members, which approaches the upper limit for effective management. The team size can grow and shrink from day to day, depending upon the workload. However, all team members must be assigned full time from the first day until the team leader releases them.

2.2.3 THE TECHNICAL SECRETARY

The team must have a technical secretary to free the technical staff from tasks that interfere with their work.

This person arranges lodging, rental cars, and air travel; and performs routine administrative coordination with the PO, the home offices of the team members, the contractors, and prepares correspondence for the team leader's signature. This job is ideal for the young officer or civil servant trainee or upward mobility person, who has been transferred from a job that is technical, but not directly software related. The technical secretary must be kept on the job from day one to the conclusion of the project. This persons duties include the preparation and proofreading of briefing slides, and the insulation of IRG members from the numerous questions and inquiries. The technical secretary should travel wherever the IRG team goes, and attend all meetings. The technical secretary was a major factor in coordinating the work effort of the referenced IRG team.

2.3 SCOPING THE EFFORT

The first function that the team must accomplish is to precisely define the scope of effort. Although the IRG is directed to look at particular problems, and to answer specific questions, the IRG itself must decide how to investigate, and what to exclude from the investigation. The team must determine how to investigate the software areas that have problems, as well as the software which will not be evaluated because it is out of scope. For instance, the referenced IRG chose to evaluate applications software, and not the operating system software, even though some of the problem symptoms appeared to be operating system failures. That decision was made after spending many days with individuals familiar with the system.

SECTION 3

3.0 CONDUCTING THE REVIEW

3.1 ORIENTATION

The IRG must quickly learn their charter, the constraints within which they must work, and the nature of the system under review. The team leader should brief all incoming members on their mission and ground rules. The PO should arrange a top-down sequence of briefings to place the system under review in context, and should make available PO technical staff members to answer the questions of the IRG. By receiving briefings, asking questions and reviewing the status of program documention, the team will be able to accurately scope the problem, and prepare for the eventual detailed analysis.

3.2 DEVELOPING EVALUATION CRITERIA

The only way an IRG can conduct a detailed software analysis is to review the existing software product and/or development methodology. During the process of scoping the problem, the IRG team also identifies sections of the software project which requires in-depth analysis. Product areas to consider include: data base design, configuration management, program listings, quality assurance, problem identification and pursuit, documentation, testing discipline, and specifications (functional, preliminary, detailed design). It is likely that more than one software area requires in-depth analysis. For example, after scoping the problem for over a week: applications software program listings, software trouble reports, configuration managememt procedures, and the data base management procedures were the specific areas of investigation chosen.

Once the TRG has scoped the effort and identified the software areas requiring a detailed analysis, the next step is to evaluate the software areas. However, this cannot be done until the team has established criteria in a 'checklist' form for evaluating the software. These criteria should be a set of standards, specifications, or procedures which the team will use as 'goodness' checks for the software. These criteria must be carefully developed, since the IRG will use them to answer the specific questions that were previously posed by the team leader and the client. The criteria must have the following features:

a) They must be practical. Allow for the status of the current software, and do only what must be done to determine exactly the software trouble spots. The software is already in trouble, so applying rigid textbook standards is not the answer.

b) Start with criteria already used for the software development-- with current standards, specifications, or procedures that the government or the contractor has applied to the software. If no criteria have been used, then the IRG will have to establish a minimum set of criteria. This minimum set should be based on fundamental design principles, rather than specifics such as structured programming, or code walk-throughs.

c) Be flexible and broad. This first cut at establishing criteria will probably not be usable as a point- by-point checklist. Instead, it will probably be more like categories of items, in outline form, to be filled in specifically just before the actual software is evaluated, and as more information is gathered about the software. Establish guidelines rather than details. The details will come later.

d) Tailor any criteria to the specific system. The system hardware, software, or operations may be specific enough so that unique criteria can be established which would apply to this system and no others.

As an example, the referenced IRG developed its own checklist, because few criteria were placed upon or used by the contractor during the software development. Because of the importance of the criteria list, the following paragraphs describe in detail the manner in which the referenced put together its list of criteria.

3.3 SOFTWARE DESIGN CRITERIA DEVELOPED BY THE IRG

Prior to any investigation of actual code or specifications, the IRG selected a list of seven broad principles which, if adhered to, would assure good design, implementation, traceability, and maintainability of a software program throughout its life cycle. The IRG did not attempt to make this list of principles comprehensive enough to include everything that could be done during software development to assure good design. The IRG also did not attempt to give a lot of detail about each design principle that was selected--that would be done later. The seven design principles were Modularity, Conventions, Documentation, Error Detection and Reporting, Interconnections, OS vs Task Responsibilities, and Testing.

These seven principles were selected based upon three factors:

a) The IRG was limited in time and personnel. A large number of design principles would dilute the IRG effort.

b) The IRG team members had differing backgrounds and level of expertise . The team consisted of members from industry and government, with experience in mainframe computers, system acquisition, research and development, maintenance, and planning. Each member approached software from different angles and emphasized some design principles more than others. However, all team members agreed that certain major design principles were absolutely necessary to assure good software development. The seven principles selected were thought to be the minimal requirements for good software design.

c) The known status of the software. The IRG team was aware that the software was going through tests, already coded, and encountering severe problems. The team knew how the software was organized into tasks. They had been briefed on the interrupt, memory management, task scheduling, and I/O management responsibilities of the software, and they were aware of the stated purpose of the IRG: to evaluate the existing applications software for its ability to pass the required tests and the software's ability to be maintained over its life. The seven design principles were selected to evaluate the software based upon its given status.

Once the seven principles were selected, a list of attributes was attached to each one, which broadly described each principle and outlined the specific items used during the investigation. These attributes were later refined into actual data collection forms.

SOFTWARE DESIGN PRINCIPLES/ATTRIBUTES

1. MODULARITY
   - Top down (branching
   - Single entrance, single exit
   - Single function
   - Interfaces
2. CONVENTIONS (against a standard from the contractor)
   - Register usage
   - Variable naming
   - Flowchart consistency
   - Protocol
   - Calling conventions
   - Coding conventions
3. DOCUMENTATION
   - C5s reflect code
   - Are modules explained
   - Adequate data/variable comments
4. ERROR DETECTION & REPORTING
   - Centralized
   - Consistency/Reliability
   - Completeness
5. INTERCONNECTIONS
   - Global/local variables
   - Parameter and data passing
   - Data Base directories/design
6. OS vs TASK RESPONSIBILITIES
   - Memory management
   - Task scheduling & management (interrupts)
   - I/O management (XORMNGR)
7. TESTING
   - How patches are handled in code/documentation
   - Test inputs/criteria for passing
   - Loading on system
   - Reporting
   - Test right things; do they pass
   - Test tools (simulators, dump, trace)
   - Failure diagnosis/correction (How easy to find problem's cause? How easy to fix?)

3.4 DATA COLLECTING AND ANALYSIS

To insure an objective analysis, hard data is needed. This requires quantification as much as possible. (Although the 'gut feel' is important, it is not the basis for million dollar decisions.)

What can be quantified? This depends upon the software product under investigation. In general, grading scales can be established for those matters requiring quality judgement by the team members. These numbers can be weighted, averaged, and aggregated, and presented to show where the team feels the software is weakest, and where of the collection forms: 'Do the code listings have adequate, relevant comments?' This is a subjective evaluation of overall maintainability which is hard to quantify. A grading scale was used with ranges from 'Completely Agree = 6' to 'Completely Disagree = 0' to quantify the analysts' perceptions. The team can also count lines of code examined, to show how much of the software the team looked at. These counts can serve as a statistical basis for later computations showing where coding problems occur, as a function of lines of code. The counts expose modules of code that are too big or too small. For example, the IRG discovered that modules exceeding 500 lines of code were of very poor quality and needed repair. The team could then project this to indicate that 100% of the large modules the team looked at needed repair, and there was a certain percentage of large modules in all the code, so that a given percentage of code needed repair. Many other methods can be used to quantify results. The point is, quantify wherever possible, or the team will be talked out of any conclusion the team may reach.

Quantification requires a set of evaluation criteria. From these criteria, a set of data collection forms can be created. Each criterion can be analyzed and expanded into point-by-point details on standardized data collection forms. As example, one of the criteria may be 'Adequate Documentation'. One of the details, which the team member can check directly in a program listing could be 'Is data in the program adequately commented?' Another documentation detail could be 'Relevant comments in the program listing'. These details should be graded and structured so that both statistical and subjective information can be gathered at the same time. A good rule is to tell the team that any statistical number not backed up by examples, or any subjective comment not backed up by numbers will not count in the final team analysis.

One problem the IRG will encounter is a learning curve. It takes a finite amount of time for even the most competent person to meet a new problem, come to grips with its scope, and do a thorough, detailed analysis. The IRG cannot expend much time for learning and preparation. The IRG team must learn and evaluate at the same time. Therefore, the data collection forms should be of varying difficulty to allow the IRG team to become familiar with the software product at the same time as collecting progressively more difficult data.

Lastly, the data collection forms should be brief. No one wants to spend more of his time analyzing the collection form than analyzing the software product. No collection form should extend over one page, even though several collection forms may be necessary for the different sets of criteria.

An example of what all this means: As mentioned before, the IRG considered four software products: the applications software program listings, the software trouble reports, the configuration management procedures, and the data base management procedures. The seven design principles were applied to these products. A brief description of this process, and descriptions of the collection forms used by the IRG members assigned to the applications software program listing, is contained in Appendix A.

3.4.1 ASSIGNING EVALUATION CRITERIA TO COLLECTION FORMS - EXAMPLE

Once the list of the IRG software design principles and attributes was finalized, a plan was devised for use of the principles during the IRG investigation. The first day of investigation would be a review of program listings for the purpose of cataloging information and familiarization with the programs. The next few days would be spent with a more detailed investigation, using a subset of the same code which was investigated during the first day. Two design principles were investigated the first day: Conventions and Documentation. Four design principles were investigated the next few days: Modularity, OS vs Task Responsibilities, Interconections, and Error Detection and Reporting. Each of these six design principles was investigated by each of the IRG team members assigned to go through the code. The seventh design principle, Testing, was assigned to another IRG team member, and was investigated independent of the code. Also, one of the attributes (data base directories/design) listed under the Interconnections design principle was investigated by another team member, independent of the code.

Of the IRG team members assigned to look at code (there were six such members), each investigated an assigned section of code and all used the same design principles and attributes. This was done to insure a standardized summary and to insure that all major sections of code were covered. This was accomplished by having each team member fill out a collection form on each task to be investigated. These forms constituted the member report on how a task program met the chosen principles.

SECTION 4

4.0 REPORTING THE RESULTS

4.1 INDIVIDUAL TECHNICAL REPORTS

Up to this point, a process and given examples of an IRG approach to analysis of a software technical problem have been described. This approach has been designed to produce an in-depth, unbiased and consistent analysis, based upon hard facts gathered from the software project under investigation. In fact, each analyst should be required to summarize his findings and report them to the team leader, using a standard final report form established by the team leader.

There is one item, however, which cannot be covered by any standard form. That item is a perception by the individual analyst on the true status of the software he analyzed. This perception is based upon many hours of analysis, with a lot of cross-checking of results between items of his section of the software project, and between other sections analyzed by the other team members. It is based on his experience and in some cases upon talking with the individuals who developed the software. These findings are not part of the data collection forms. However, this perception is valuable, and should be part of the standard summary. If the analyst is asked to give his opinion of the software, he will more than likely support it with examples as best he can. Therefore, the team leader must require this viewpoint to be expressed, and it should be included, unedited, as part of the final report.

4.2 FINAL REPORT FORMAT

The final report should be written in two volumes. Volume I should be an executive summary giving a high level view of the team's activities, conclusions, and recommendations. Volume II should be a technical report, containing the large amount of technical data, detailed analysis, and individual reports that were produced during the IRG.

4.2.1 VOLUME I - EXECUTIVE SUMMARY

The purpose of Volume I of the final report is to present the IRG approach, findings, and recommendations. Very briefly, the following outline was used by the IRG.

a) Introduction. This describes the system under investigation and the organizations and people responsible for establishing the IRG.

b) Purpose. This describes the questions to be answered by the IRG, as well as a brief description of the approach taken to answer the the questions.

c) Scope. This describes what is included and excluded from the IRG investigation.

d) Team Composition. This states the organizations the team members belong to, their relationship to the program under investigation, and their familiarity with the software.

e) Approach. This describes the criteria and collection forms used to evaluate the software. Also, it describes the specific items of the software product that received an in-depth analysis.

f) Technical Summary. The extent of the IRG analysis and the severity of the problems associated wth each evaluation criteria is described. It gives the percentage of software that was reviewed. The percentage should be by analyst as well as a total for the team. A short paragraph should be written which summarizes the severity of the problems associated with each criteria the team used. Each paragraph should describe the criteria used, the applicability of the criteria within the software, and the resulting problems and recommendations for each criteria.

g) Conclusions. This answers the questions described under the Purpose section of the executive summary report. It catagorizes all the problems, and relates the severity of the problems to each question. Technical recommendations should be made on which areas of the software should be corrected.

h) Recommendations. This section emphasizes the technical recommendations of the Conclusion section. It should describe what must be done to implement corrective action. This recommendation should estimate the impact on the final software product (quality, schedule, performance, etc.) of implementing, as well as not implementing, the IRG technical recommendations.

4.2.2 VOLUME II - TECHNICAL SUMMARY

The purpose of Volume II of the final report is to provide the support for the conclusions and recommendations expressed in Volume I. Volume II describes in very great detail the approch taken by the IRG, and the data collected during the analysis. Very briefly, the following outline was used by the IRG.

a) Introduction, Purpose, Scope, Team Composition, and Approach. These sections are identical to Volume I. This is to make Volume II a stand-alone document.

b) Technical Summary. This section should specify the detailed analysis conducted by the IRG. The overview should describe the assignments to each team member, and the number of days each member spent on each assignment. The collection forms should be described in detail. The purpose and type of information required by each form should be described. A technical analysis should be presented. This is a summary of the findings of each collection form extracted from all the collection form summaries created by the team members. The last section of this technical summary should be additional comments and recommendations by each analyst. These comments should be extracted from the analysts' individual summaries, and presented alongside the analysts' names. It is important that these additional comments not be summarized, because they deal with findings not specifically asked for in the collection forms.

The Volume II report also included the following information as appendices.

c) Team Composition (Appendix I). The team composition should include: 1)the mailing address of each team member, 2)a short resume of software development experience of each team member, and 3)a weekly record of time spent on the IRG in hours by each team member. The team leader should prepare a team data sheet for each member to supply his resume, address, and work hours.

d) Persons Contacted (Appendix 2). The IRG should list, by organization and individual, each person contacted throughout the course of the investigation.

d) Briefing Slides (Appendix 3). The IRG will produce findings, conclusions, and recommendations which will be briefed to the organizations responsible for establishing the IRG. Those briefing slides should be a part of the technical report.

f) Daily Memoranda of Activities (Appendix 4). Part of the team leader's responsibility will be to record each day's activities. This is necessary, because the IRG should have a history to trace the development of its activities and conclusions. That daily record should be part of the technical report.

g) Completed Collection Forms (Appendix 5). The collection forms filled out by the IRG members contain the raw data which form the basis of the IRG technical summaries, findings, conclusions, and recommendations. These collection forms should be included, as they were written, directly into the technical report. Also, a blank copy of all the collection forms used should be included.

h) Individual Technical Reports (Appendix 6). The technical reports should be presented exactly as written by the individual team members. These technical reports represent a summary of each individual's analysis, as perceived by the individual. (The technical summary for Volume II is actually a condensed digest of these team members' findings.)

i) Acronyms and Abbreviations (Appendix 7). A list of all abbreviations and acronyms used in Volumes I and II of the final report should be provided.

SECTION 5

5.0 SUMMARY

A software IRG investigation of a troubled ongoing system acquisition program is an intense, challenging job. The IRG must produce an accurate analysis of the problems and prepare technical recommendations for solutions in a very short time, usually under suspicious, unfriendly conditions on a crash basis, while on TDY. The IRG success depends upon excellent people, specific questions, limitations of scope, and good planning. This paper has presented a picture of one successful IRG. It could be beneficial to other IRG teams in the future to help the planning portion of their effort.

CONVENTIONS AND DOCUMENTATION COLLECTION FORM

The first day, the team investigated two design principles: Conventions and Documentation. Originally, the IRG team planned to analyze the coding conventions against contractor standards. This plan was aborted after determining that the contractor's standards were limited and would not have provided information necessary to make a comparison. The actual collection form was titled "Standards and Conventions".

The first day was viewed as an opportunity to become familiar with the software as well as to gather useful statistical information. To accomplish this, a mechanical-type analysis which required counts, yes/no answers, and little analysis was performed. Both Conventions and Documentation fell into this category and were combined into one collection form. The attributes listed for each principle were refined and a total of ten items was finally used for the analysis. The items required quantitative and subjective ratings by each team member. Eight of the ten items could be answered by looking at program listings, the other two by reference to design specifications. Not all questions were relevant to all tasks, for instance, some design specifications were out of date and both assembly and HOL coded programs were investigated.

The first day was designed to examine a large volume of code. The IRG team believed this would indicate trends and severity of bad software practices and provide a statistical level of confidence for any conclusions which would be made during the next few days.

Date:________________

STANDARDS AND CONVENTIONS

Analyst:________________

CPCI:________________

CPC:________________

TASK PROGRAM:________________

LANGUAGE:________________

CONFIG.
CONTROL: VERSION ________________

SYSGEN DATE:________________

PATCHES
TRAILING?________________

LINES OF CODE ANALYZED________________

1. Task programs are adequately described by <u>Header Comments</u> or in the appropriate C-5	A	B	C	D	E	F
2. Code listings have adequate, relevant <u>comments</u>.	A	B	C	D	E	F
3. <u>Data</u> in the program is adequately <u>commented</u>.	A	B	C	D	E	F
4. Registers, constants are defined as EQU statements.	A	B	C	D	E	F
5. In-line data literals are not used. (i.e., AND R3, 0177400)	A	B	C	D	E	F
6. Program counter relative jumps are not used. (i.e., JMP $+7)	A	B	C	D	E	F
7. Conventions for common subroutines, common data, and IPP data packets are recognizably standard. (i.e., are parameters to sub-routines always passed with registers or variables in consistent ways?)	A	B	C	D	E	F
8. Conventions for OS service calls are consistent.	A	B	C	D	E	F
9. C-5 specs and code listings reflect one another accurately. (i.e., flows and corresponding code can be located and tracked with comparative ease)	A	B	C	D	E	F
10. Flowcharts honor conventions consistently. (i.e., standard symbols, flow conventions, branch conditions)	A	B	C	D	E	F

<u>A</u>. Completely agree; <u>B</u>. Strongly agree; <u>C</u>. Generally agree;
<u>D</u>. Generally disagree; <u>E</u>. Strongly disagree; <u>F</u>. Completely disagree.

MODULARITY OF TASK PROGRAMS COLLECTION FORM

The IRG considered modularity to be a good indicator of how well each task program was structured. The individual attributes of the Modularity design principle were tailored to specifics about our particular software (such as memory page boundandaries and HOL loops).

A total of eight items was stressed in the Modularity collection form. All eight of these items required only numerical or yes/no responses. The intent was to force quantification of the analysis. Also, each team member was instructed to provide comments, references, and diagrams to back up any numbers which reflected bad task program modularity. In this manner, each team member could support his subjective analysis of the task modularity.

DATE: ____________

MODULARITY OF TASK PROGRAMS

ANALYST: ____________

CPCI: ____________

CPC: ____________

TASK PROGRAM: ____________

CONFIG. CONTROL: VERSION ____________

LANGUAGE: ____________

SYSGEN DATE: ____________

LINES OF CODE ANALYZED ____________

No. Loops (Task Internal)	____________
No. ENTRY/EXIT POINTS	____________
No. DSPL UNCONTROLLED GOTOs	____________
No. FUNCTIONS BOUND TO A TASK	____________
SIZE OF CODE (Memory Use)	____________
No. Functional Relationships (Intra-Memory Page Transfers between functions)	____________
No. Recognizable Functions in a memory page	____________
Task Program extend over a single page	____________

INTERCONNECTIONS COLLECTION FORM

The IRG considered interconnections to be a good indication of how standard the connections were and how difficult it would be to understand and use any commonality between and within tasks. The individual attributes of the Interconnection design principle were expanded slightly and used for the collection form. Although not specifically a part of the collection form, certain error detection and reporting schemes would be reported, as applicable.

A total of six items was stressed in the Interconnections collection form. These items required both numerical and subjective responses. It was an easy matter to obtain the numerical responses, but it was more difficult to analyze the goodness, visibility, and appropriateness of the task interconnections. The collection form required team members to support their subjective responses with in-depth analysis and specific examples.

DATE: ________________

INTERCONNECTIONS

ANALYST: ________________

CPCI: __________

CPC: __________

TASK PROGRAM: ____________________ CONFIG. CONTROL: VERSION ________________

LANGUAGE: ____________________ SYSGEN DATE: ______________

PATCHES TRAILING? ______________

LINES OF CODE ANALYZED ________________

No. DYNAMIC MEMORY VARIABLES	____________	A B C D E F
No. GLOBAL VARIABLES	____________	A B C D E F
No. CALLING SEQUENCES	____________	A B C D E F
No. DIFFERENT CALLING SEQUENCES (includes Return Conventions and includes register vs. variable data passing)	____________	A B C D E F
VISIBILITY OF CALLING SEQUENCES	A B C D E F	
No. COMMON ROUTINES USED	____________	A B C D E F

A. Completely agree; B. Strongly agree; C. Generally agree;
D. Generally disagree; E. Strongly disagree; F. Completely disagree.

OS VS TASK RESPONSIBILITIES COLLECTION FORM

The IRG studied this topic because of reports received by the team prior to the detailed investigation. These reports indicated inconsistency and inappropriate division of responsibility between functions of the operating system (OS) and individual tasks. The IRG team considered this analysis to be a good indicator of how well control responsibility was managed by indivdual task programs and whether programs were consistent in their approach to OS interfaces.

A total of eight items was stressed in the OS vs Task Responsibilities collection form. These items required both numerical and subjective responses. A major part of the analysis included the Error Detection and Reporting design principle, since error indicators and error checking are integral parts of most OS calls. Consistency and traceability were the major items the collection form stressed, and both a numerical and descriptive response to the collection form items was required.

DATE:______________

OS vs. TASK RESPONSIBILITIES ANALYST:______________

(Include Common Subroutines)

CPCI:__________

CPC:__________

TASK PROGRAM:________________ CONFIG. CONTROL: VERSION________________

LANGUAGE:________________ SYSGEN DATE:________________

PATCHES TRAILING?________________

LINES OF CODE ANALYZED________________

MEMORY MGMT.
No. DIFFERENT WAYS OF HANDLING ____________

SHORT DESCRIPTION OF EACH WAY

ONE FOR ONE ALLOCATE/RELEASE	YES/NO				circle	
TRACEABLE (EASE)	A	B	C	D	E	F
TASK SCHEDULING UNDERSTANDABILITY (DESCRIPTION, HOW WHY)	A	B	C	D	E	F

CLASS II ERROR DETECTION & REPORTING

No. DIFFERENT METHODS ____________

No. ERROR RETURNS CHECKED ____________

DESCRIBE EACH METHOD

A. Completely agree; B. Strongly agree; C. Generally agree;
D. Generally disagree; E. Strongly disagree; F. Completely disagree.

TECHNICAL SUMMARY

"NOTE: Try to keep your summary to one double-spaced typed page for each Design Principle analyzed (i.e. Modularity of Task Programs, Interconnections, etc.)."

- -

Analyst: Date:

"NOTE: The following format applies to each Design Principle writeup."

1-4. Design Principle Name:

a. Data Summary: (mostly amounts, counts, etc.)

b. Examples: (at least two specific examples where a Design Principle violation can be found in the code). Also describe the examples and give the code reference.

c. Quantify what you looked at (i.e. scope, relative goodness, badness, etc.).

d. Bottom line feeling: must be rewritten, redesigned, should be rewritten, etc., relate to testability and maintainability.

e. Final recommendations.

5. Additional Comments: (concerns for things found that do not relate to the other four categories).

TEAM DATA SHEET

Full Name:

Rank/Grade:

Title:

Organization:

Full Mailing Address:

Phone Number:

Formal Education:

Short Statement of S/W Development Experience:

TIME SPENT ON IRG (IN HOURS)

WEEK OF:

Team Member	1 Jan	8 Jan	15 Jan	22 Jan	TOTAL

GRAND TOTAL ____